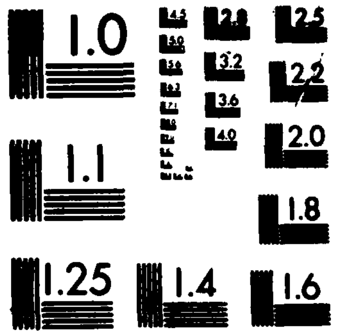
AD-A183 615 SIMPLIFYING DECISION TREES(U) MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB J R QUINLAN DEC 86 AI-M-930 N00014-85-K-0124 1/1

UNCLASSIFIED F/G 12/9 NL

END
9-87
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

12

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AIM-930	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SIMPLIFYING DECISION TREES		5. TYPE OF REPORT & PERIOD COVERED AI-Memo
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) J.R. QUINLAN		8. CONTRACT OR GRANT NUMBER(s) N00014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE DECEMBER 1986
		13. NUMBER OF PAGES 16
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Induction, decision trees, pruning, production rules, knowledge acquisition

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Abstract: Many systems have been developed for constructing decision trees from collections of examples. Although the decision trees generated by these methods are accurate and efficient, they often suffer the disadvantage of excessive complexity that can render them incomprehensible to experts. It is questionable whether opaque structures of this kind can be described as knowledge, no matter how well they function. This paper discusses techniques for simplifying decision trees without compromising their accuracy. Four methods are described, illustrated, and compared on a test-bed of decision trees from a variety of domains.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

AI Memo No. 930 December, 1986

Simplifying Decision Trees

J. R. Quinlan[1]

Abstract: Many systems have been developed for constructing decision trees from collections of examples. Although the decision trees generated by these methods are accurate and efficient, they often suffer the disadvantage of excessive complexity that can render them incomprehensible to experts. It is questionable whether opaque structures of this kind can be described as knowledge, no matter how well they function. This paper discusses techniques for simplifying decision trees without compromising their accuracy. Four methods are described, illustrated, and compared on a test-bed of decision trees from a variety of domains.

Acknowledgements: Much of the work described in this report was carried out at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

[1] Permanent address: School of Computing Sciences, New South Wales Institute of Technology, Sydney, N.S.W. 2007, Australia.

1. Introduction

Since people began building knowledge-based systems, it has become painfully obvious that the ability to function at an expert level in some task domain does not necessarily confer a corresponding ability to articulate this know-how. The knowledge for most early and many current expert systems has been amassed by an interview process in which a *knowledge engineer* interacts with a domain expert to extract and refine a set of *rules*. This process can be taxing for all concerned because the expert, as Waterman (1986) puts it,

> "has a tendency to state [his] conclusions and the reasoning behind them in general terms that are too broad for effective machine analysis ... the pieces of basic knowledge are assumed and combined so quickly that it is difficult for him to describe the process."

Consequently, the productivity of the interview method is usually low. This led Feigenbaum (1981) to identify knowledge acquisition as the "bottleneck" problem in building knowledge-based systems.

One way around this bottleneck, long advocated by Donald Michie (1983) and others, uses inductive methods to extract general rules from concrete examples. The expert is not asked to articulate his skill but instead to provide a framework of important concepts in the task domain, augmented perhaps by a collection of tutorial examples; the hard work is carried out by a suitable induction engine. Most researchers in Machine Learning will be familiar with Meta-DENDRAL and its synthesis of chemical knowledge (Buchanan and Mitchell, 1978) and with AQ11's results on soybean diagnosis (Michalski and Chilausky, 1980). The feasibility of this inductive approach to knowledge acquisition has also been confirmed in several industrial projects, such as British Petroleum's recent successful construction of a 2500-rule expert system for the design of hydrocarbon separation vessels in just one man-year (*Expert Systems User*, August 1986, pp16-19).

Many current commercial induction packages (including *Ex-Tran*, *RuleMaster* and *1st-Class*) express the derived rules in the form of *decision trees*. From the standpoint of execution efficiency this is a simple and economical representation, but the trees can become complex and thus opaque (Michie, 1986). If a decision tree that measures up very well on the performance criterion is nevertheless totally incomprehensible to a human expert, can it be described as *knowledge*? Under the common-sense definition of this term as material that might be assimilated and used by human beings, it is not, in just the same way that a large program coded in assembly language is not knowledge.

This paper examines four methods for improving the intelligibility of decision trees and thereby making them more knowledge-like. Three of the methods involve *pruning* the decision tree by replacing one or more subtrees with leaves, while the remaining method reformulates the decision tree as a set of production rules. Section 2 introduces the methods and illustrates their operation with respect to a small but real example. Section 3 presents an empirical comparison of the methods using sets of decision trees from six task domains.

2. Methods for Simplifying Decision Trees

Induction algorithms that develop decision trees view the task domain as one of *classification*. The underlying framework consists of a collection of *attributes* or properties which are used to describe individual *cases*, each case belonging to exactly one of a set of *classes*. Attributes may be either continuous or discrete. A case's value of a continuous attribute is always a real number while its value of a discrete attribute is one of a small set of possible values for that attribute. In real-life tasks it is also important to recognize that a case may have *unknown* values for one or more of the attributes.

A *decision tree* may be either a leaf identified by a class name, or a structure of the form

$$\begin{array}{ll} C_1: & D_1 \\ C_2: & D_2 \\ \cdots & \cdot \\ C_n: & D_n \end{array}$$

where the C_i's are mutually exclusive and exhaustive logical conditions and the D_i's are themselves decision trees. The set of conditions involves only one of the attributes, each condition being

$$A < T \text{ or } A > T$$

for a continuous attribute A, where T is some threshhold, or

$$A = V \text{ or } A \text{ in } \{V_i\}$$

for a discrete attribute A, where V is one of its possible values and $\{V_i\}$ is a subset of them. To improve legibility, the non-leaf subordinate decision trees above will be indented when the trees are printed.

Such a decision tree is used to classify a case as follows. If the tree is a leaf, we simply determine the case's class to be the one nominated by the leaf. If the tree is a structure, we find the single condition C_i that holds for this case and continue with the associated decision tree. The only complexity arises when the value of the attribute appearing in the C_i's is unknown. In this eventuality we explore all the decision trees associated with the structure and combine their findings with weights proportional to the estimated probability of the associated condition being satisfied. (Quinlan, 1986) discusses the procedure in more detail.

Figure 1 shows such a decision tree for the diagnosis of hypothyroid conditions with classes {*primary hypothyroid*, *secondary hypothyroid*, *compensated hypothyroid*, *negative*}. Some attributes such as *TSH* and *FTI* are continuous and have real values, while attributes like *thyroid surgery*, with possible values $\{t, f\}$, are discrete. To classify a case with this tree, we would first enquire whether the value of *TSH* was greater than 6.05. If

```
TSH < 6.05:
|   T4U measured = t: negative (1918)
|   T4U measured = f:
|   |   age > 43.5: negative (58)
|   |   age < 43.5:
|   |   |   query hypothyroid = f: negative (41)
|   |   |   query hypothyroid = t: secondary hypothyroid (1)
TSH > 6.05:
|   FTI < 64.5:
|   |   thyroid surgery = f:
|   |   |   T3 < 2.3: primary hypothyroid (51)
|   |   |   T3 > 2.3:
|   |   |   |   sex = M: negative (1)
|   |   |   |   sex = F: primary hypothyroid (4)
|   |   thyroid surgery = t:
|   |   |   referral source = SVI: primary hypothyroid (1)
|   |   |   referral source = <other>: negative (2)
|   FTI > 64.5:
|   |   on thyroxine = t: negative (32)
|   |   on thyroxine = f:
|   |   |   thyroid surgery = t: negative (3)
|   |   |   thyroid surgery = f:
|   |   |   |   TT4 < 150.5: compensated hypothyroid (120)
|   |   |   |   TT4 > 150.5: negative (6)
```

Figure 1: Sample Decision Tree

the value was below this threshhold we would continue with the decision tree commencing with $T4U\ measured = t$, while a value above this threshhold would lead us to the decision tree headed $FTI < 64.5$. In either case we would continue in similar fashion until a leaf was encountered.

The set of cases with known classes from which a decision tree is induced is called the *training set*. Other collections of cases not seen while the tree was being developed are known as *test sets* and are commonly used to evaluate the performance of the tree.

This paper focusses on simplifying decision trees, not with the inductive methods used to construct them in the first place. Various ways of developing trees from training sets may be found in (Breiman, Friedman, Olshen and Stone, 1984), (Kononenko, Bratko and Roškar, 1984) and (Quinlan, 1986).

2.1 Cost-Complexity Pruning

Breiman *et al* (1984) describe a two-stage process in which a sequence of trees $T_0, T_1, ..., T_k$ is generated. T_0 is the original decision tree and each T_{i+1} is obtained by replacing one or more subtrees of T_i with leaves until the final tree T_k is just a leaf. The second stage evaluates these trees and selects one of them as the final pruned tree.

Consider a decision tree T used to classify each of the N cases in the training set from which T was generated, and let E of them be misclassified. If $L(T)$ is the number of leaves in T, Breiman *et al* define the ***cost-complexity*** of T as the sum

$$\frac{E}{N} + \alpha \times L(T)$$

for some parameter α. Now, suppose we were to replace some subtree S of T by the best possible leaf. In general, the new tree would misclassify M more of the cases in the training set but would contain $L(S) - 1$ fewer leaves. This new tree would have the same cost-complexity as T if

$$\alpha = \frac{M}{N \times (L(S) - 1)}$$

As before, let T_0 be the original tree. To produce T_{i+1} from T_i we examine each non-leaf subtree of T_i to find the minimum value of α above. The one or more subtrees with that value of α are then replaced by their respective best leaves.

To illustrate the process, consider the decision tree of Figure 1. This was generated from 2514 cases, where the number in parentheses after each leaf shows how many of these cases are covered by that leaf.[2] Consider the subtree

```
T4U measured = t: negative (1918)
T4U measured = f:
|   age > 43.5: negative (58)
|   age < 43.5:
|   |   query hypothyroid = f: negative (41)
|   |   query hypothyroid = t: secondary hypothyroid (1)
```

The vast majority of cases at the leaves of this subtree are of class *negative* which is clearly the best leaf. If the subtree were replaced by the leaf *negative* the new tree would misclassify the lone non-*negative* case, so M is 1. The new tree would also have three fewer leaves, giving a value for α of 0.00013 at which the cost-complexity of the original and modified trees would be equal. This is the lowest such value for any subtree, so the tree T_1 would be formed by replacing this subtree as above.

[2] The counts do not sum to 2514 because cases with unknown values of tested attributes cannot be associated with any one leaf and are therefore not included.

```
TSH < 6.05: negative (2018)
TSH > 6.05:
|   FTI < 64.5: primary hypothyroid (62)
|   FTI > 64.5:
|   |   on thyroxine = t: negative (32)
|   |   on thyroxine = f:
|   |   |   thyroid surgery = t: negative (3)
|   |   |   thyroid surgery = f:
|   |   |   |   TT4 < 150.5: compensated hypothyroid (120)
|   |   |   |   TT4 > 150.5: negative (6)
```

Figure 2: Decision Tree After Cost-Complexity Pruning

The second stage of this process abandons the cost-complexity model and attempts to select one of the T_i's on the basis of reliability alone. We cannot assess this simply from the proportion of cases in the original training set that are misclassified. Whatever induction algorithm was employed has almost certainly built the original tree to fit the training set and thus the error rate on these cases would be expected to understate the error rate on unseen cases. We therefore assume some test set containing N' cases and use each T_i to classify all of them. Let E' be the minimum number of errors observed with any T_i, with the standard error of E' being given by

$$se(E') = \sqrt{\frac{E' \times (N' - E')}{N'}}$$

The tree selected is the smallest T_i whose observed number of errors on the test set does not exceed $E' + se(E')$.

In this example, a test set containing 629 cases gave a sequence of eight trees, T_0 being the original tree and T_7 the leaf *negative*. The selected tree was T_4 which appears in Figure 2. This tree is indeed a great deal simpler than the original and would qualify as 'knowledge' under the most stringent criterion. Notice that the class *secondary hypothyroid*, which is represented by just a single case in the training set, has sensibly been omitted.

Nevertheless, cost-complexity pruning raises several problematic issues. First, it is unclear why the particular cost-complexity model used above is superior to other possible models such as the product of error rate and number of leaves. Secondly, it seems anomalous that the cost-complexity model used to generate the sequence of subtrees is abandoned when the best tree is selected. Finally, the procedure requires a test set distinct from the original training set; the authors show, however, that a cross-validation scheme can be employed to generate these estimates at the time the original tree is constructed, but at the expense of a substantial increase in computation.

```
TSH < 6.05: negative (2018)
TSH > 6.05:
|   FTI < 64.5:
|   |   thyroid surgery = f: primary hypothyroid (59)
|   |   thyroid surgery = t:
|   |   |   referral source = SVI: primary hypothyroid (1)
|   |   |   referral source = <other>: negative (2)
|   FTI > 64.5:
|   |   on thyroxine = t: negative (32)
|   |   on thyroxine = f:
|   |   |   thyroid surgery = t: negative (3)
|   |   |   thyroid surgery = f:
|   |   |   |   TT4 < 150.5: compensated hypothyroid (120)
|   |   |   |   TT4 > 150.5: negative (6)
```

Figure 3: Decision Tree After Reduced Error Pruning

2.2 Reduced Error Pruning

Rather than form a sequence of trees and then select one of them, a more direct procedure suggests itself as follows. We again assume a separate test set, each case in which is classified by the original tree. For every non-leaf subtree S of T we examine the change in misclassifications over the test set that would occur if S were replaced by the best possible leaf. If the new tree would give an equal or fewer number of errors and S contains no subtree with the same property, S is replaced by the leaf. The process continues until any further replacements would increase the number of errors over the test set.

Using the same example of Figure 1 and the same test set as before, reduced error pruning generates the tree shown in Figure 3.

As with cost-complexity pruning, this process generates a sequence of trees. Its rationale is clearer, though, since the final tree is the most accurate subtree of the original tree with respect to the test set and is the smallest tree with that accuracy. The disadvantages of the method are, first, that it again requires a separate test set and second, that parts of the original tree corresponding to rarer special cases not represented in the test set may be excised.

2.3 Pessimistic Pruning

When the original tree T is used to classify the N cases in the training set from which it was generated, let some leaf account for K of these cases with J of them misclassified. As observed before, the ratio J/K does not provide a reliable estimate of the error rate of that leaf when unseen cases are classified, since the tree has been tailored to the training set. A more realistic error rate might be obtained using the continuity correction for the binomial distribution (Snedecor and Cochran, 1980, pp. 117ff) in which J is replaced by $J + 1/2$.[3]

Let S be a subtree of T containing $L(S)$ leaves and let ΣJ and ΣK be the corresponding sums over the leaves of S. A more pessimistic view of S is that it will misclassify $\Sigma J + L(S)/2$ out of ΣK unseen cases, where the standard error of this number of misclassifications can be determined as before. If S were replaced by the best leaf, let E be the number of cases from the training set that it misclassifies. The pessimistic pruning method replaces S by the best leaf whenever $E + 1/2$ is within one standard error of $\Sigma J + L(S)/2$. All non-leaf subtrees are examined just once to see whether they should be pruned but, of course, sub-subtrees of pruned subtrees need not be examined at all.

To illustrate the idea we return to the subtree of Figure 1 that commences with the condition *T4U measured* $= t$. As before, ΣK is 2018, $L(S)$ is 4, ΣJ is 0, so the estimate of the number of errors due to S is 2.0 with standard error 1.41. If the subtree is replaced by the leaf *negative* it will give one error, so E is 1. Since $1 + 1/2 < 2.0 + 1.41$, pessimistic pruning would indeed replace this subtree. Repeating this evaluation on all subtrees of T gives a pruned tree identical to that of Figure 2.

This method has two advantages. It is much faster than either of the preceding methods since each subtree of T is examined at most once. Unlike these methods, it does not require a test set separate from the cases in the training set from which the tree was constructed.

2.4 Simplifying to Production Rules

This form of simplification does not give a smaller decision tree at all but instead develops an 'equivalent' set of *production rules*, a representation medium widely used in expert systems (Winston, 1984). The process has two stages: individual production rules are first generated and polished, and then the rules produced are evaluated as a collection.

Whenever a decision tree is used to classify a case, a path is established between the top of the tree and one of its leaves. In order for the case to reach that leaf, it must have satisfied all the conditions along the path. For example, any case that is classified as *negative* by the last leaf of the decision tree in Figure 1 must satisfy all the conditions

$$\begin{array}{l} TSH > 6.05 \\ FTI > 64.5 \\ \textit{on thyroxine} = f \\ \textit{thyroid surgery} = f \\ TT4 > 150.5 \end{array}$$

[3] This makes the unsurprising assumption that $J/K < 0.5$.

Every leaf of a decision tree thus corresponds to a production rule of the form

$$\textbf{if } X_1 \wedge X_2 \wedge \ldots \wedge X_n \textbf{ then } \textit{class } c$$

where the X_i's are conditions as before and c is the class of the leaf.

Merely rewriting a tree as the collection of these equivalent production rules would not represent any simplification at all. Instead, the first stage examines each production rule to see whether it should be generalized by dropping conditions from its left-hand side. Let X_i be one of the conditions and consider those cases in the training set that satisfy all the other conditions in the rule. With respect only to these cases, the relevance of X_i to determining whether a case belongs to class c (given that the other conditions are satisfied) can be summarized by the 2 × 2 contingency table

	class c	not class c
satisfies X_i	sc	$s\bar{c}$
does not satisfy X_i	$\bar{s}c$	$\overline{sc}$

where sc is the number of these cases that satisfy X_i and belong to class c, $s\bar{c}$ is the number that satisfy X_i but belong to some class other than c, and so on. Fisher's exact test (Finney, Latscha, Bennett and Hsu, 1963) can then be invoked to assess the probability that the division by X_i arises merely from chance or, in other words, the significance level at which we can reject the hypothesis that X_i is irrelevant to whether a case belongs to class c.[4] Each X_i is examined in turn to find the one that has the least relevance to classification and, unless the hypothesis that this X_i is not significant can be rejected at the 1% level or better, the condition is discarded and the process repeated.

Consider the rule above. When the training cases that satisfy all conditions other than the first are examined, the table for the condition $TSH > 6.05$ comes out to be

	class *negative*	not class *negative*
$TSH > 6.05$	6	0
$TSH < 6.05$	154	0

which shows that this condition is entirely irrelevant. On the other hand, the table of cases satisfying all conditions other than the last is

[4] I am indebted to Donald Michie of the Turing Institute for making me aware of this test and its advantages over the approximate χ^2 test.

	class *negative*	not class *negative*
$TT4 > 150.5$	6	0
$TT4 < 150.5$	0	120

which is significant at better than the 0.1% level. Repeated application of the above process reduces the original rule to one with a single condition

if $TT4 > 150.5$ **then** *class negative*

The final step in this first stage is to estimate a *certainty factor* for the simplified rule, using a device similar to that of pessimistic pruning. If the left-hand side of a rule is satisfied by V cases in the training set, W of which belong to the class indicated by the right-hand side, the certainty factor of the production rule is taken as $(W - 1/2)/V$. In the example above, the training set contains 246 cases that match the left-hand side, all of them being class *negative*, so this rule's CF is 99.8%.

Note that we need not develop one rule for each leaf of the decision tree. Some leaves give rise to identical rules while other leaves generate vacuous rules from which all conditions have been dropped. The number of rules is generally smaller than the number of leaves.

The second stage of this process looks how well the rules will function as a set. This aluation depends on the way in which the rules will be used. A simple strategy has been adopted here:

> To classify a case, find a rule that applies to it. If there is more than one, choose the rule with the higher certainty factor. If no rule applies, take the class by default to be the most frequent class in the training set.

For each rule in turn, we now determine how the remaining rules would perform on the training set if this rule were omitted. If there are rules whose omission would not lead to an increased number of errors classifying the cases in the training set, or would even reduce it, the least useful such rule is discarded and the process repeated.

Continuing the example, the decision tree of Figure 1 is reduced by this method to just three rules:

if $TSH < 6.05$ **then** *class negative* [99.9%]

if *thyroid surgery* $= f \wedge$
$TSH > 6.05 \wedge$
$FTI < 64.5$ **then** *class primary hypothyroid* [97.5%]

$$\begin{array}{lll} \textbf{if} & \textit{on thyroxine} = f \wedge & \\ & \textit{thyroid surgery} = f \wedge & \\ & TSH > 6.05 \wedge & \\ & TT4 < 150.5 \wedge & \\ & FTI > 64.5 & \textbf{then} \quad \textit{class compensated hypothyroid} \ [99.6\%] \end{array}$$

As with pessimistic pruning, this method does not require a set of test cases apart from the original training set. In its current implementation it is the slowest of the four tree-simplifying methods. The method should be able to be improved by adopting a more sophisticated condition-elimination strategy than the simple hill-climbing approach used above, and by employing a better production rule interpreter.

2.5 Other Methods

The four methods of simplifying decision trees certainly do not exhaust all possibilities. The cross-validation method of Breiman *et al* (1984) has already been mentioned. Kononenko *et al* (1984) present an information-based heuristic used in their ASSISTANT system, but this is now being changed to another form of cross-validation (Lavrač, Mozetič and Kononenko, 1986). I have previously experimented with a form of pruning based on the path lengths in the decision tree and observed error rates (Quinlan, Compton, Horn and Lazarus, 1986).

3. Empirical Comparison

The performance of a simplification method can be assessed in terms of the clarity and accuracy of its final product. Ideally, the pruned decision tree or set of production rules should be much more comprehensible than the original decision tree but should not be significantly less accurate when classifying unseen cases.

To test how well the methods of the previous section measure up to these two criteria, they were compared using decision trees developed for six task domains. For each domain, the available data was shuffled, then divided into a training set containing approximately two-thirds of the data and two equal-sized test sets. This division was carried out so as to make the proportion of cases belonging to each class as even as possible across the three sets. The training set was used to induce ten decision trees for the domain. Each simplification method was applied to each tree and the resulting classifier evaluated on both test sets.

The six domains include both real-world tasks and synthetic tasks constructed to provide some particular challenge. They are

- Diagnosis of hypothyroid conditions (***Hypothyroid***): This domain has been encountered in the running example of the previous section. The data comes from the archives of the Garvan Institute of Medical Research, Sydney, and covers all 3772 thyroid assays carried out by Garvan's clinical laboratory between January and November 1985. The data uses seven continuous and sixteen discrete attributes with quite high rates of missing information – values of four of the attributes are unknown in more than 10% of the cases. The 3772 cases, each belonging to one of four classes, were split into a training set of 2514 and two test sets of 629. This domain is a good starting point because it uses 'live' data from which, warts and all, extremely accurate classifiers can be constructed.

- Discordant assay results (*Discordant*): This domain is taken from the same Garvan data, this time looking to detect anomalous combinations of thyroid hormone values. There are two classes and the 3772 cases were divided as above. The percentage of discordant cases is very low (about 1.5%) and, in contrast with the first domain, the decision trees generated from this training set perform comparatively poorly on unseen cases.

- Recognizing faulty digits (*LED Digits*): The third domain comes from (Breiman *et al*, 1984). Imagine a seven-element representation for a decimal digit such as is commonly found on LED or LCD displays. Each element of a faulty display is subject to a 10% random error, i.e. with probability 0.1 its correct status is inverted. The data consists of 3000 randomly-generated cases, each described in terms of the seven binary attributes, with ten equiprobable classes. The training set contains 2000 cases, the test sets 500 each. This artificial domain is interesting because it tests the ability of the simplification methods to deal with the complex decision trees commonly obtained from noisy training sets.

- Assessing consumer credit applications (*Credit*): The data for this domain was provided by a large bank. Each case concerns an application for credit card facilities described by 9 discrete and 6 continuous attributes, with two decision classes. The 690 cases making up the data are divided into a training set of 460 and two test sets of 115. Some discrete attributes have large collections of possible values (one of them has 14) resulting in broad, shallow decision trees. This data is also both scanty and noisy, giving decision trees that are extremely complex and not very accurate on unseen cases.

- King and rook versus king and knight (*Endgame*): This domain from a chess endgame seeks to decide whether the rook's side can capture the opposing knight and/or checkmate in 3 ply. Positions are described by 39 binary attributes, with all possible board positions giving rise to 551 distinct cases. This domain models an idealized noise-free environment with no missing information in which the accuracy of the decision tree depends only on the completeness of the training set. Here the training set contains 367 cases, the test sets 92 cases each.

- Probabilistic classification over disjunctions (*Prob-Disj*): The last domain is an artificial one designed to model tasks in which only probabilistic classification is possible and which contains explicit disjunctions. There are ten boolean attributes a_0 through

	Original Decision Trees	Cost-Complexity Pruning	Reduced Error Pruning	Pessimistic Pruning	Production Rule Form
Hypothyroid	23.6 nodes	11.4 nodes	14.4 nodes	11.0 nodes	3.0 rules
Discordant	52.4 nodes	11.8 nodes	12.4 nodes	13.6 nodes	1.8 rules
LED Digits	92.2 nodes	45.6 nodes	59.0 nodes	56.0 nodes	15.8 rules
Credit	248.0 nodes	9.7 nodes	26.3 nodes	32.5 nodes	7.8 rules
Endgame	88.8 nodes	51.0 nodes	55.6 nodes	62.6 nodes	11.6 rules
Prob-Disj	190.0 nodes	30.4 nodes	43.0 nodes	42.6 nodes	4.2 rules

Table 1: Average Size Before and After Simplification

a_9 and the criterion used to generate the data can be expressed as: if $a_0 \wedge a_1 \wedge a_2$ or $a_3 \wedge a_4 \wedge a_5$ or $a_6 \wedge a_7 \wedge a_8$ then the class is Y with probability 0.9, N with probability 0.1; otherwise, the class is N with probability 0.9, Y with probability 0.1. (The remaining attribute a_9 is irrelevant.) Because the class of a case is determined probabilistically, no classification procedure can achieve more than 90% accuracy on this task. Six hundred cases with random values for each attribute were generated and classified as above. Of these, 400 are used as the training set, leaving test sets of 100 cases each.

The results of these experiments are summarized in the following tables. The effectiveness of the simplification methods in reducing the size of the original decision trees is shown in Table 1, each entry being the average over the ten decision trees in that domain. As a general observation, all the methods achieve significant simplification in all domains. Cost-complexity pruning tends to produce smaller decision trees than either reduced error or pessimistic pruning, especially in the *Credit* domain. While the complexity of decision trees and sets of production rules cannot be compared directly, it would appear that the last method achieves the greatest reduction overall, its advantages being particularly noteworthy in the *Prob-Disj* domain.

The other side of the coin is the effect of simplification on classification accuracy. Table 2 shows the results in each domain of using the ten original decision trees and their simplified counterparts to classify cases in the two test sets, expressed as the average percentage of misclassifications over each set. Perhaps surprisingly, the simplified trees on the whole are of superior or equivalent accuracy to the originals, so pruning has been beneficial on both counts. Note, though, that both the cost-complexity and reduced error methods

	Original Decision Trees	Cost-Complexity Pruning	Reduced Error Pruning	Pessimistic Pruning	Production Rule Form
Hypothyroid					
Test 1	0.3%	0.4%	0.3%	0.5%	0.3%
Test 2	0.8%	0.7%	0.8%	0.6%	1.0%
Discordant					
Test 1	1.6%	1.1%	1.0%	1.0%	1.1%
Test 2	2.1%	1.6%	1.7%	1.5%	1.5%
LED Digits					
Test 1	30.0%	29.9%	27.8%	28.8%	31.3%
Test 2	27.9%	28.7%	28.0%	27.4%	28.3%
Credit					
Test 1	20.2%	14.4%	12.9%	15.8%	15.2%
Test 2	21.0%	17.1%	17.4%	16.4%	17.8%
Endgame					
Test 1	11.8%	13.8%	10.0%	13.1%	11.1%
Test 2	10.5%	13.4%	11.6%	12.1%	7.3%
Prob-Disj					
Test 1	17.0%	14.2%	10.1%	14.0%	10.0%
Test 2	18.4%	17.2%	17.8%	15.8%	10.0%

Table 2: Average Error Rates on Test Sets

have "seen" ιhe first training set in performing their respective simplifications. The slight superiority of reduced error pruning, coupled with the fact that cost-complexity pruning produces smaller trees, suggests that the latter may be slightly over-pruning. Despite *not* having seen the first test set, the performance of pessimistic pruning is marginally better than cost-complexity pruning averaged over all domains. Simplification to production rules, though, scores pretty clear wins in the last two domains. In the *Prob-Disj* domain in particular, this can be explained by observing that disjunctive concepts tend to scatter cases from some disjuncts throughout the decision tree. Pruning the tree is unable to re-collect these cases, but simplification of rules can.

One further possibility has been explored. There is no obvious way to merge distinct decision trees, so pruned trees from different originals cannot be combined to form a composite tree that reflects the various strengths of its components. No such limitation applies to the production rule representation, though, because the union of sets of rules is itself a set. This line of thought led to a final experiment in which, for each domain, the rule sets produced from all ten decision trees were amalgamated and the collection winnowed as before. The composite rule set was then used to classify all cases in the test

	Number	Error Rates	
	of Rules	*Test 1*	*Test 2*
Hypothyroid	3	0.3%	1.0%
Discordant	2	0.6%	1.4%
LED Digits	23	28.2%	25.8%
Credit	11	13.0%	15.7%
Endgame	12	9.8%	5.4%
Prob-Disj	4	10.0%	10.0%

Table 3: Error Rates of Composite Rule Sets

sets. The results in Table 3 show that these composite sets of production rules are both compact and accurate classifying mechanisms, matching or outperforming the best of all other methods on nine of the twelve test sets.

4. Conclusion

The intention of this paper has been to investigate methods for simplifying decision trees without compromising their accuracy. The motivation behind this drive towards simplicity is the desire to turn decision trees into knowledge for use in expert systems.

Four methods have been discussed, all of which managed to achieve significant simplification when put to the test on sets of decision trees from six task domains. This simplification was often coupled with an actual improvement in classification accuracy on unseen cases. Two of the four methods needed a separate set of test cases in order to carry out the simplification and, since these did not perform noticeably better than the remaining two methods, the requirement of additional test data is a weakness. The last method, in which decision trees are reformulated as sets of production rules, has proved especially powerful.

Acknowledgements

I am grateful to the Garvan Institute of Medical Research for providing access to the thyroid data, and to Les Lazarus and Paul Compton in particular for their help. This work has been supported in part by grants from the Australian Research Grants Scheme and the Westinghouse Corporation.

References

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1984), *Classification and Regression Trees*, Belmont: Wadsworth.

Buchanan, B.G., and Mitchell, T.M. (1978), Model-directed learning of production rules, in *Pattern Directed Inference Systems*, (D.A. Waterman and F. Hayes-Roth, Eds.), Academic Press.

Feigenbaum, E.A. (1981), Expert systems in the 1980s, in *State of the Art Report on Machine Intelligence*, (A. Bond, Ed.), Maidenhead: Pergamon-Infotech.

Finney, D.J., Latscha, R., Bennett, B.M. and Hsu, P. (1963), *Tables for Testing Significance in a 2 × 2 Contingency Table*, Cambridge University Press.

Kononenko, I., Bratko, I., and Roškar, E. (1984), Experiments in automatic learning of medical diagnostic rules, Technical Report, Jozef Stefan Institute, Ljubljana, Yugoslavia.

Lavrač, N., Mozetič, I. and Kononenko, I. (1986), An experimental comparison of two learning programs in three medical domains, *Proceedings of ISSEK Workshop 86*, Turing Institute, Glasgow.

Michalski, R.S. and Chilausky, R.L. (1980), Learning by being told and learning by examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis, *International Journal of Policy Analysis and Information Systems 4*, 2.

Michie, D. (1983), Inductive rule generation in the context of the Fifth Generation, *Proceedings of the Second International Machine Learning Workshop*, University of Illinois at Urbana-Champaign.

Michie, D. (1986), Current developments in expert systems, *Proceedings of the Second Australian Conference on Applications of Expert Systems*, Sydney.

Quinlan, J.R. (1986), Induction of decision trees, *Machine Learning 1*, 1.

Quinlan, J.R., Compton, P.J., Horn, K.A. and Lazarus, L. (1986), Inductive knowledge acquisition: a case study, *Proceedings of the Second Australian Conference on Applications of Expert Systems*, Sydney.

Snedecor, G.W. and Cochran, W.G. (1980), *Statistical Methods* (7th edition), Iowa State University Press.

Waterman, D.A. (1986), *A Guide to Expert Systems*, Addison-Wesley.

Winston, P.H. (1984), *Artificial Intelligence* (2nd edition), Addison-Wesley.